

.....

IBM

Architecting a Digital FX and Animation Studio Infrastructure



*IBM Digital Media: Solutions for the
Media and Entertainment Segment*

OR: Coloring Peta-pixels Blue

Veronika Megler, Sr. IT Architect
George Dolbier, Sr. IT Architect
IBM Sales and Distribution
Small and Medium Business, Emerging and Competitive Sales, Digital Media

V1.0

Copyright and Trademarks

Copyright © 2005 by IBM Corporation. All rights reserved.

IBM, IBM (logo) and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries.

Linux is a registered trademark of Linus Torvalds.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Disclaimer

This work represents the view of the authors, and does not necessarily represent the view of IBM.

This paper is based on laboratory tests undertaken in a laboratory environment. Results in particular customer installations may vary based on a number of factors, including workload and configuration in each particular installation. Therefore, the above information is provided on an AS IS basis. The WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED. Use of this information is at the user's sole risk.

Table of Contents

Disclaimer	ii
Table of Contents	iii
Introduction	1
The Business: A Brief Overview	1
Step One: Understanding Flash Blue's Needs.....	2
The Big Animal Picture	1
Content Creation/Editing Stations	3
Render Farm.....	6
Render Engines.....	7
Render Management.....	8
Centralized Storage	11
NAS	12
SAN	12
iSCSI.....	12
Network File Systems	13
Advanced File System(s).....	13
Choosing a Storage and File Systems Design.....	14
So What Will We Do Here?	15
Networking.....	16
Putting it All Together	17
Sizing.....	20
How many workstations will we need?	20
How big should our render farm be?	20
How do we size the individual render nodes?	22
How much storage will our studio need?	22
What storage bandwidth will our studio need?	22

How do I size the centralized storage system? 23

Sizing the File System Servers..... 25

What networking bandwidth will our studio need? 25

The Final Picture.....27

Introduction

I see pixels, lots and lots of pixels...

The Digital Special Effects (SFX) and computer animation industry, a sub-segment of the Media and Entertainment industry, is exploding world wide. There are a number of reasons for this, and they involve the merging of technology and imagination.

Over the past decade or two, the number and complexity of special effects and animations being created has dramatically increased, along with customers' expectations. Remember how earthshaking the first Star Wars movie seemed? Now, when you go back and see it again, it seems unbearably primitive.

More recently, even within the Lord of the Rings trilogy, the number of visual effects shots started at 540 in the first film and roughly doubled for each of the next two movies. Industrial Light and Magic (ILM) in San Rafael, Calif., constantly faces this pressure. "In the first Jurassic Park movie, we did 75 shots. Now, with a Star Wars movie, every shot has some effect in it," for a total of 2,000 to 2,500 shots per film, says Chief Technology Officer Cliff Plumer.¹ Greg Butler, digital computer graphics supervisor at Weta, says: "Film audiences expect visual effects to keep blowing them away. The only way this is possible is through the constant upgrading of our infrastructure."

Now, you may think to yourselves "how much of this can there be, one Shrek, and one Pixar movie every other year, and a couple of kids shows on TV. What's the big deal?" Consider all the TV commercials that contain some form of digital effect (that's 95% of all commercials). All major motion pictures use some form of digital effect. Most popular is digital lighting and after production effects. Now you have a good idea of the size of the US market. There is also a market in Latin America, India, China, Korea, Japan, Northern Europe, France, Russia and Australia. In many of these markets animation is as, if not more, acceptable as a communication medium as film. (Think Japanese "anime", now being seen at a theater near you, and being reprised in Kill Bill 2.)

All this technology growth has led to an increase in the technology infrastructures needed. Initially, as in most technology fields, growth occurred "organically", and equipment was added as it was needed. But now, companies are investing several million dollars in a studio infrastructure – and are realizing the cost of growth and upgrading without an architecture in mind. So, the emphasis on an architectural approach is emerging.

This paper will describe how to architect the infrastructure that a studio needs to support their increasingly technology-supported business. We'll follow a particular example, for an animation studio we'll call "Flash Blue", from business requirements through to the physical architecture they will need.

The Business: A Brief Overview

Companies that create digital effects, 2D and 3D animation for film, television, and print are referred to by many names: animation houses, animation studios, post production studios, preproduction studios, and the list goes on. Sometimes there is a notable difference in capability denoted by the name: For example, an animation studio provides script creation and

¹ "Blades, Camera, Action!" Robert L. Mitchell, ComputerWorld, October 4, 2004
<http://www.computerworld.com/hardwaretopics/hardware/server/story/0,10801,96284,00.html>

story boarding, vs. a post production studio, which provides technical effects and editing. Sometimes the differentiation is in customer set only.

A studio is a company that distributes film or television content. They have money, or the ability to borrow money. At the highest level these companies are outsourcers. The service they provide is derived from the artists and technicians working for the company.

There are only a small number of “Tier 1” animation and production studios in the world, and they are all names we know (PDI/Dreamworks, Lucasfilm/ILM, Pixar, Disney Animation, WETA). “Tier 2” companies exist in every geography, supporting the regional film industries, such as Hollywood, Bollywood, Toronto, etc. Then, a set of contract studios exist to support the other tiers, effectively forming a third tier.

Our example here, Flash Blue, is a contract studio that is making a bid to become a tier 2 company. They have a green-lighted project for which they’ve negotiated a distribution contract with a major distributor. It’s their first true 3D project, and now they need to start ramping up quickly.

Step One: Understanding Flash Blue’s Needs

So how does a notion in someone's head make it on screen?

A studio approaches an animation house with a project proposal. After some minor negotiations the studio will pay for some initial character development, and initial storyboarding, which may include animated storyboards. This work may also include several seconds' high quality test animation. This initial work is presented to the studio for project approval, the studio will make recommendations, and the animation house will do some rework.

This back and forth occurs for some time. If the studio likes what they see, the animation house will be required to submit a project budget, and proposed schedule to the studio. More negotiations occur. Once the budget, schedule and initial story are approved, the studios will “green light” the entire project; alternatively, the studio kills the project and everybody walks away.

Here is the important point. When the studio green lights the project, they cut a big fat check to the animation house. (Well, actually it is more like the studio extends a line of credit, or provides some form of access to funds, but you get the idea). In most cases the animation house needs to build out an infrastructure PRONTO, because as soon as the project is “green lighted”, the clock starts ticking down toward the production release date. This is where Flash Blue is right now.

So our animation studio with a green lighted project is a company in panic. This company has to ramp up from nothing (the infrastructure they built on a shoestring prior to green light), to full length feature animation production infrastructure. This is a compute environment that would make NASA, AMES, and the national supercomputer center envious. For example, the installation at Weta Digital is ranked on the Top 500 supercomputer list as one of the world’s largest supercomputer sites.²

Now, most film industry projects follow a fairly standard literary form:

² “Blades, Camera, Action!” Robert L. Mitchell, ComputerWorld, October 4, 2004
<http://www.computerworld.com/hardwaretopics/hardware/server/story/0,10801,96284,00.html>

- A movie or feature (project), like a play, is divided into “acts” (typically 3)
- Acts are divided into “scenes” (again, as in a play)
- Scenes are divided into “shots” (typically between 3 and 9). “In film, a **shot** is a continuous strip of motion picture film, created of a series of frames, that runs for an uninterrupted period of time. It generally portrays a subject, though a blank screen can also be considered a shot. Shots are filmed with a single camera and are of variable duration. .”³
- Shots are divided into “cuts”. Each cut is a single visual unit, typically a single viewpoint; they can include pan, zoom, still, combined with music, sound and dialog.
- For each “cut”, multiple “takes” are created. A “take” is an artist's attempt to realize the director’s vision for the “cut”.
- All takes completed in a day are accumulated and turned into “digital dailies”. A “daily” is reviewed by the Direction and/or editorial staff in a regularly scheduled daily meeting.

A scene can be compared to a paragraph, a shot can be compared to a sentence, with a cut being a words and each frame a letter. These subdivisions make it easier to manage the work required to put together the project.

There are far more companies that do commercials and digital special effects. These companies are more concerned about "SHOTS" rather than scenes. The differences can be boiled down to two simplistic statements:

1. Commercials are divided up into shots, or cuts and that's it (no higher level).
2. The work required for digital special effects rendering is often more resource intensive than typical cinematic 3D rendering (the "quality" of each frame is lower, but the amount of work, tweaking, and rework that goes into each frame is much higher) and

So, by understanding these industry-based measures for Flash Blue, we can begin to collect the business information we’ll need in order to understand the size of the project they have in mind, and therefore, estimate the infrastructure they’ll need.

Flash Blue will be making a single 3D movie, 90 minutes long. They have a script, although they know from past experience that they’ll change it “in flight”. Here are their estimates.

Project Measure	Flash Blue Estimate
Number of Projects	1
Number of Artists	80-90; 100 at peak
Duration of Production Phase	9 months
Avg. / Expected Length of Project	90 minutes
Avg. Number of Acts/Project	4-5
Avg. Number of Scenes/Act (or Project)	90 scenes in the movie
Avg. Number of Cuts/Scene	2,000 cuts for the movie
Expected Number of Takes/Cut	5-7

³ From Wikipedia: http://en.wikipedia.org/wiki/Shot_%28film%29

Project Measure	Flash Blue Estimate
Frames/Sec.	30; some cuts will be developed at 60 frames/sec, when interlaced, for high-speed action scenes.

Table 1: Flash Blue's Planned Project

The goal of a picture is to realize a director's vision. To accomplish this, editorial and direction are key. Essentially every creative step goes through an editorial and directorial phase. The actual workflow of a studio is highly customized studio to studio, and actually can change at a rather rapid pace. These steps also vary slightly depending on whether the studio is creating a completely animated feature (the steps shown here), or whether they are, for example, adding special effects or characters to a movie that also involves filmed actors.

Here is an example of one studio's production workflow.

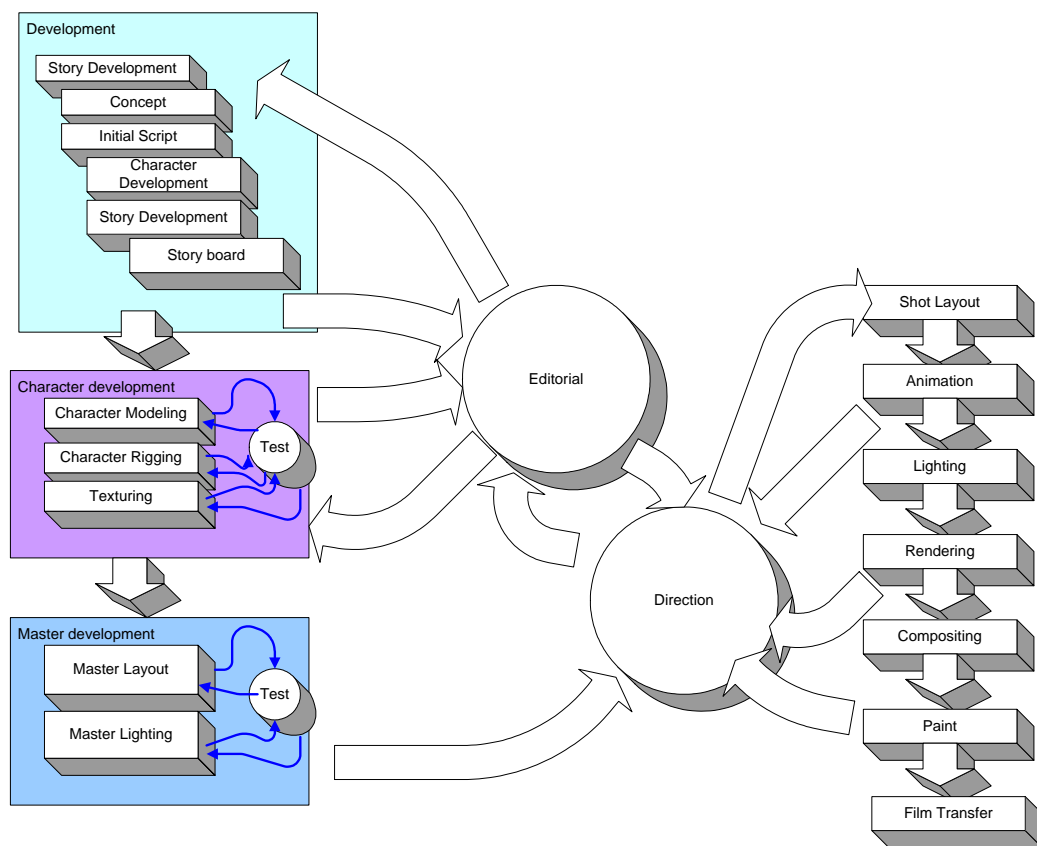


Figure 1: Example of Studio Workflow

Note the number of loops, and the number of steps various types of content go through. This is why content management with workflow, and version control, is so important. Much of this workflow can easily be automated IF, and only IF you can centrally store all the assets. Animation studio and post production architectures bear a striking resemblance to many High Performance Computing architectures. At a high level, the architecture for genomic research could be picked up and dropped into any animation studio.

Each of the steps shown in the studio workflow involves different types of work. Here is further explanation of each of these steps:

- Shot layout is often hand drawn animation in comic, or story board format. This is often considered “pre-production”.
- “Animation” consists of creating each character and associated motion. Each character will be created by several people specializing in different aspects, such as face, hands/feet, torso, hair, and so on.
- Lighting and “rigging” provide the framework allowing assets to move and look convincing.
- Rendering is the mathematical process of combining lighting, motion, and character shape information into high resolution, high color images. Backgrounds/foregrounds and other elements may be rendered separately from the moving characters.
- Combining rendered characters with rendered foregrounds and other elements is called compositing. This typically manual task is now being highly automated by software such as “Final Cut Pro” from apple and “After Effects” from Adobe.
- The final act after editing is completed is to transfer the digital images to film. Typically this is done by one of a small handful of film industry companies, such as Technicolor or Kodak. This is considered “post-production”.

Note also that each of these steps involves separate types of work – all of them time-consuming, and each of them likely to be performed using different applications. Because the artists are highly-trained and relatively expensive, the studio will often purchase the applications they say will be most time-effective (and therefore cost-effective) to achieve the desired results or that they are trained in, rather than mandating a set of applications that the artists are expected to use. As you’ll see, this can create interesting (!) complexities for the architecture, as we progress towards the physical incarnation.⁴

So, one of the first things we need to do is to collect the list of the applications this studio is going to use.

Here is Flash Blue’s answer to that question.

Task	Main Application to be Used	Comments
Shot layout	Premier, Virtual Dub, Photoshop	Select based on work convenience
Animation	Maya	Can consider 3ds Max
Lighting	Maya	Depends on animation tools
Rendering	Maya, Mental Ray, Renderman	Use multiple applications according to the need of the shot. Development for specific functionality is

⁴ Another good reference for the workflow, although it’s a little older, is “Linux Helps Bring Titanic to Life”, Daryl Strauss, 1998-02-01, <http://www.linuxjournal.com/article/2494>

Task	Main Application to be Used	Comments
		possible.
Compositing	Shake, Inferno, After Effects, Digital Fusion	
Paint	Photoshop (mainly) Paint (for backgrounds)	
NLE	Avid Adrenaline 1EA (proxy editing), Fire (final)	
Film transfer	Pass Data or HD tape to development room (SGI,TIFF)	

Table 2: Flash Blue's Applications

As you can see, the resources required to make a full feature or more than one concurrent TV series is HUGE. And there are only sooo many animators on the planet. To get an example of just how rare this resource is, let me provide this comparison. Someone with a technical mind can become productive on Linux in 3 to 6 months, and you can make a Linux expert in 3 years. Well, someone who is going to be an animator has to be good at drawing, and drawing things that move, and drawing things that move convincingly. It takes 11 to 12 years to train such an artist.

As a result, any technology that can be shown to reduce the skilled staff required, or make them more productive, has value to this market. All these companies need the same technologies: production content creation software, servers, high performance networks, inexpensive storage, and high performance workstations.

Here's a diagram that represents the applications and flows we're talking about here. Note that this begins the transition to the technical views of their needs. Note also that there are a couple of functions that we'd expect to find that they specifically tell us they will not need:

- They are creating a purely animated feature, and so will not need the power of an automatic ingest station. The small amounts of ingest that they are planning will be handled by their Non-Linear Edit station as an adjunct to its regular work.
- They are outsourcing the creation of the audio track, and so will not need to ingest, edit or transform audio either. The audio track will be synchronized with the video in their non-linear editor, again as an adjunct to their regular work. This simplifies life for us, since we will not need to deal with audio tools and their card support and preferences.

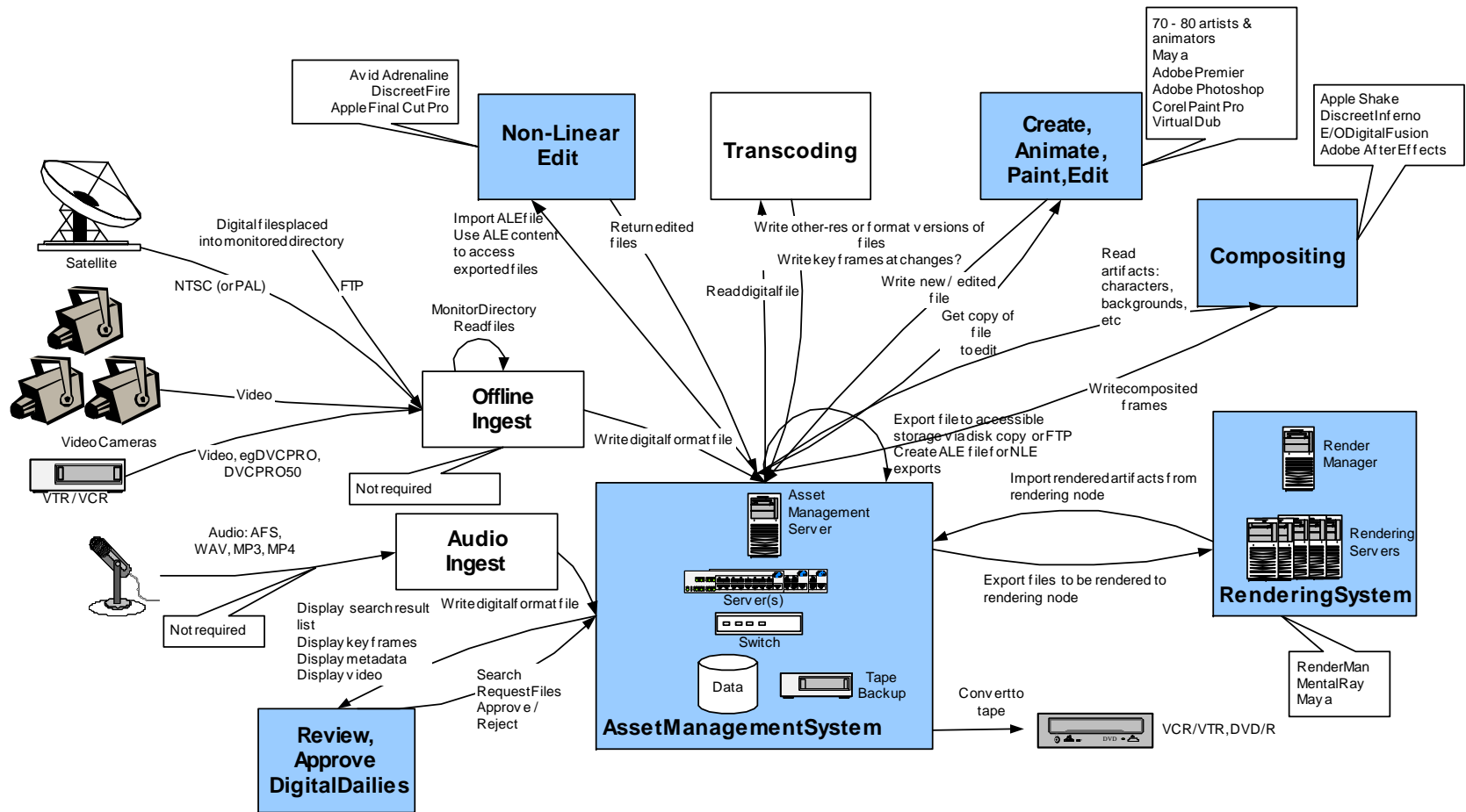


Figure 2: Flash Blue, Functional Architecture

The Big Animal Picture

The actual infrastructure used to perform this workflow and these various functions is made up of a few major subsystems, as can be seen in Figure 2:

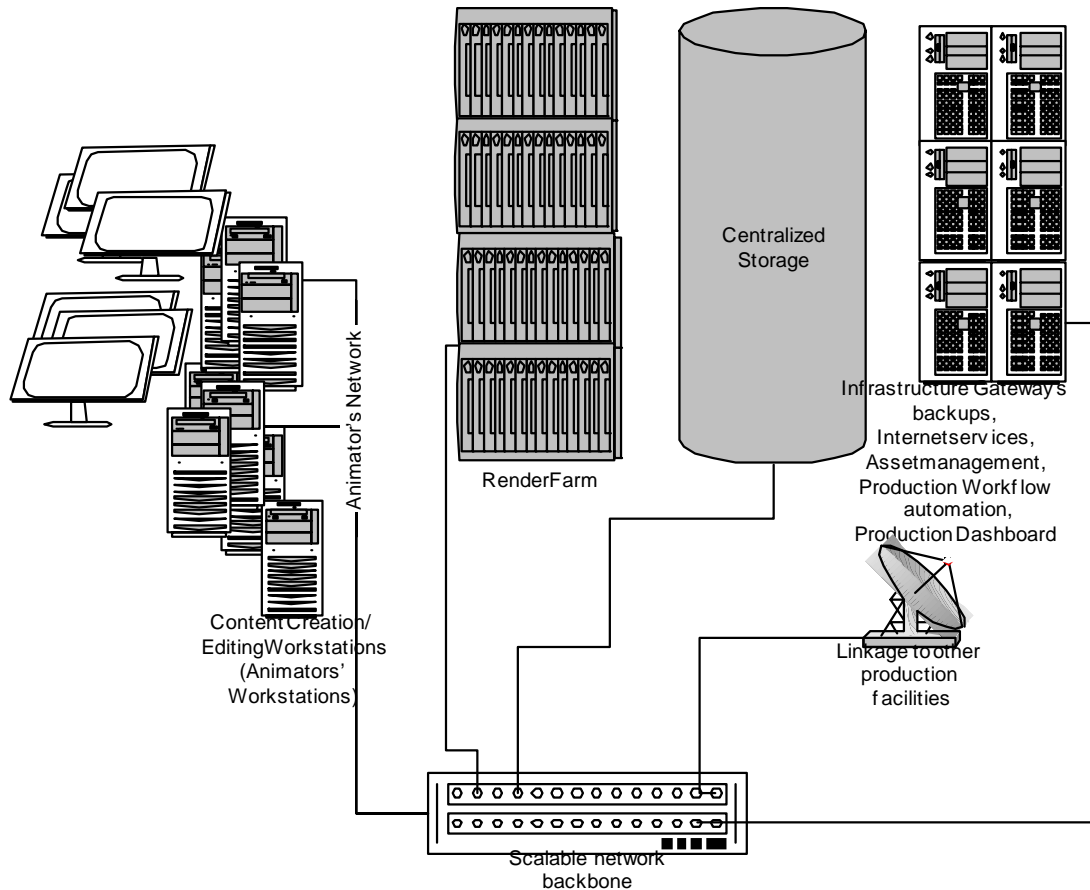


Figure 3: The Big Animal Picture

- Content creation / editing workstations: In cinematic and broadcast content creation there are two major uses for workstations: As digital content creation (DCC) or animation workstations, and as editing workstations.
- Render farm: A render farm in general is a collection of networked computers, called nodes, all rendering some part or all of a frame of animation.
- Centralized storage: Animation requires a lot of data, and there are often multiple people who need to access it. Centralized storage, provided it is affordable, fits this need.
- Network: Network bandwidth is critical to the timely delivery of any production in this market. Unfortunately adequate bandwidth has been out of the price reach for most studios.

But, how do we make this high-level architecture specific to Flash Blue?

1. Let's first capture the basic questions we need to answer, for each of the components in our picture.
2. We'll then identify the answers to these questions, ignoring, for the moment, the issue of sizing.
3. Then, we'll go back and work out, for each of our components, how many we'll need – which will drive us to another level of detail towards our implementation.

For the workstations and render farm, we need to know the applications, the operating system, and the hardware – three interdependent questions. For the centralized storage, we'll need to know the storage architecture, file system, and access mechanism – again, three interdependent questions. For the network, we'll need to calculate the bandwidth.

Here's our big animal architecture, annotated with our key questions.

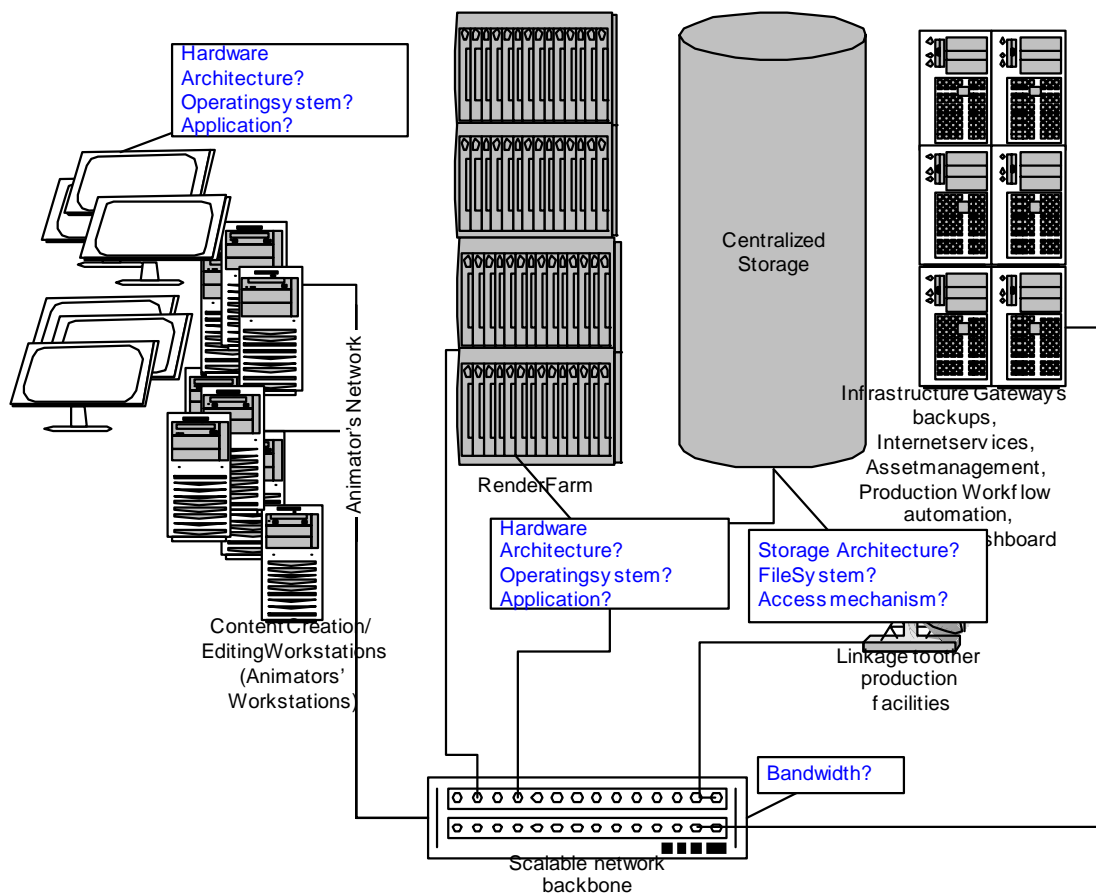


Figure 4: Questions We'll Need to Answer

So let's discuss each of these components, and start to answer these questions. Let's start with a more detailed description for each of the major components in the big animal picture: content creation / editing workstations, the render farm, centralized storage and the network.

Content Creation/Editing Stations

Fundamentally, what we are talking about here are desktops, and in some cases laptops. The devil is in the details, as they say.

The process of creating rich digital content, whether it be video for TV, cinematic film, or graphics for interactive entertainment (such as games), can be broken down into two basic activities:

- Content creation: this is the process of creating 2D or 3D images, animations, backgrounds, textures etc., etc.
- Content editing. Editing is the processing and sequencing of motion sequences into some final form. There are two types of “editors”, the traditional analog linear editors, and the much more productive Non Linear Editors (NLEs). Analog editing is done with tape decks; Non Linear Editing is performed on workstations.

Some organizations would also create a third major classification, called compositing. Compositing is the processes of creating a final visual form by the combination of various elements or layers. Here we consider compositing as a form of editing.

The content creation workstations are where the artist’s applications primarily run. Content creation and content editing are very graphics I/O intensive processes.

For **animation workstations**, the **content creation** applications rule this industry. One of the first things we must understand about our studio is what tools they use to create the content. The exact workstation configuration needed will be driven by the list of applications we extracted, above.

So, let’s take a look again at the applications Flash Blue has specified, and what system requirements they have.

Vendor	System Requirements (Examples, Extract from vendor websites)
Adobe Photoshop http://www.adobe.com	Microsoft Windows 2000 or XP
Adobe Premier http://www.adobe.com VirtualDub http://www.virtualdub.org/	Premier: Microsoft Windows XP VirtualDub: Windows only, freeware. There’s an experimental version compiled specifically for the 64-bit modes of the AMD Athlon 64, AMD Athlon FX, AMD Opteron, and Intel Xeon EM64T CPUs
Alias Maya http://www.alias.com	Windows® XP Professional or Windows ® 2000 Professional (service pack 2 or higher) <ul style="list-style-type: none"> ▪ Red Hat Linux 9.0 and Red Hat Enterprise Linux 3.0 WS ▪ SUSE Linux 9.1 ▪ IRIX 6.5.15 (ATTENTION Users of the IRIX Operating System) ▪ Apple® Mac® OS X 10.3 or higher
Mental Images Mental Ray http://www.mentalimages.com	32-bit platforms: Apple Mac G4/G5, HP/UX for HPPA 2.0, IBM AIX, Linux for Intel x86 and AMD x86, SGI Irix, SunOS, Windows 2000/XP for Intel x86 and AMD x86 64-bit platforms: HP Tru64 UNIX (Alpha), HP/UX for IA64 Itanium and HPPA 2.0, IBM AIX (Power 4 and up), Linux for AMD x86-64 Opteron and Intel IA64 Itanium, SGI IRIX64, Windows XP for Intel IA64 Itanium

Vendor	System Requirements (Examples, Extract from vendor websites)
Pixar Renderman https://renderman.pixar.com/	The RenderMan® Artist Tools™ and RenderMan® Pro Server™ are available on the following platforms: RenderMan® Pro Server™ 12.0 is supported on the following hardware and operating systems: Apple® Mac® OS X 10.3.1 (or higher), Red Hat™ Linux® 7.x, 9.0 (32-bit), SuSE® 9 Pro, RHEL AS/WS (x86-64), Windows® XP Pro, 2000 Pro (SP2+) The RenderMan® Artist Tools™ 6.0 is supported on the following hardware and operating systems: (MTOR requires valid Maya install) Red Hat™ Linux® 7.x, 9.0, SGI® IRIX® 6.5, Windows® XP Pro, 2000 Pro (SP2+) Comments: Pixar also sells Alfred, a render manager that is optimized to work with Renderman. It can also use artists' workstations as part of the renderfarm when they are idle.
Apple Shake http://www.apple.com	Apple Mac G5, Mac OS X Linux
Discreet Inferno http://www4.discreet.com/inferno/	High performance 4-8 CPU SGI® ONYX® 350 with support for IR4 graphics subsystem with real-time 8/10-bit 4:4:4 ITU-R 601 standard definition and SMPTE 292M HDTV I/O. Support for 2K 10- and 12-bit film.
Adobe After Effects http://www.adobe.com	Intel® Pentium® III or 4 processor (multiprocessor recommended) Microsoft® Windows® 2000 with Service Pack 4 or Windows XP Professional or Home Edition with Service Pack 1 For the Render Engine: System requirements are the same as application system requirements. Only specific graphics cards are supported.
Eye Online Digital Fusion http://www.eyeonline.com/	Windows 2000 / XP Dual Processor AMD or Intel
Corel Paint Shop Pro http://www.corel.com	Microsoft® Windows® 98SE, 2000 (SP4), ME, XP Microsoft® Internet Explorer 6.0 or later
Avid Adrenaline 1EA http://www.avid.com/products/dna/adrenaline/index.asp	HP or Dell Precision workstation; or, Apple Power Mac G5/G4 Windows XP Professional with Service Pack 2 on PC systems Mac OS X 10.3.4, 10.3.5, 10.3.6, and 10.3.7, Mac on Macintosh systems
Discreet Fire http://www4.discreet.com/fire/	SGI® Onyx 350 with up to 8 CPUs and InfiniteReality 4 graphics subsystem
Apple Final Cut Pro http://www.apple.com/finalcutstudio/finalcutpro/	Macintosh computer with 350MHz or faster PowerPC G4 or G5 processor and AGP graphics card Mac OS X v10.3.2 or later

Table 3: Applications Current O/S Support

EEK! At the minimum, it seems that we'll have Windows 2000 / XP, Apple Mac, and SGI workstations; with options to run some applications on Linux, if we wish. We have a predominance of Intel, with occasional support of AMD. No matter what we do, we'll have a heterogeneous workstation environment.

Let's take our list of options, and sort them according to the following logic:

- Use a non-proprietary option whenever possible.

- For heavy-duty, long-running applications, choose the Linux version, if available. Our experience is that Linux rendering is more stable.

Here's the result of our sorting:

Windows XP	Linux	"Other"
Adobe PhotoShop	Alias Maya	Discreet Inferno
Adobe Premier	Mental Images Mental Ray	Discreet Fire
Adobe After Effects	Pixar Renderman	Apple Final Cut Pro
Virtual Dub	Apple Shake	
Eye Online Digital Fusion		
Corel Paint Shop Pro		
Avid Adrenaline		

Table 4: O/S Selected

Now, we can address the question of CPU architecture. Many of the Windows and Linux applications formally support both Intel and AMD Opteron-based processors.

Once we choose a CPU type, we should stick with it. There have been articles published about the fact that color values will be calculated differently by different CPU types.⁵ For this reason whichever CPU we choose for the desktop workstation, we should choose the same CPU for our render farm and NLE. This will ensure consistent color values are calculated during all stages of the animation workflow.

Creating or editing content generally needs high performance, often 3D accelerated, workstations. The workload placed on the workstation varies slightly between editing and content creation functions. Editing video is very, very I/O intensive, so performance of the disk and network components is critical. Content creation is not as I/O intensive but it is extremely interactive, and so the performance of the video and input systems is critical. For these reasons, we need a high-end, powerful workstation with high-end graphics. Given that, we'll pick a dual-Opteron AMD-based workstation, such as the IBM Intellistation A Pro.

Non-Linear Editing workstations (here, the workstations that will run Apple Final Cut Pro, Avid Adrenaline, Discreet Fire), are a breed above the average content creation workstation. Editing stations need to be able to read and write several concurrent video streams in real time, while processing them. This is not a task often left to commodity desktop computers. The I/O and functional requirements of these tasks are still beyond the reach of the average PC. Here we are limited in our workstation choices by the vendor support, so we'll end up with the SGI and Apple G4/G5 systems they suggest. In general, since these systems are used for color correction for the final target environment rather than creating new colors, having the same CPU architecture is less critical.

Non-Linear Editing systems are designed solely to provide the ability to edit video in real time. Their connection to the rest of the environment is typically through a high speed

⁵ For example: a long thread on <http://vbulletin.newtek.com/archive/index.php/t-17798.html>

network. Real-time editing of video requires access to high performance disk; for this reason NLE systems are typically connected to either direct attach storage or through a SAN.

In fact, if workstations were able to run fast enough, a workstation is all an artist would want. However, as soon as workstations get faster, artists come up with more extensive effects that they would like to create – and so the need for a render farm never quite goes away. Today, most 2D animation can be performed on workstations ... But the moment the animators move to 3D animation, the need for a render farm inevitably follows.

Render Farm

A single frame can easily be rendered on an animator's workstation. Entire scenes can be rendered on the animator's workstation, but the process is so intensive that the animator can do nothing else at the same time. So, the *business* reason for a render farm is to off-load a render to a system that is not being used by an animator.

The basics are fairly simple to grasp. Special effects and animation are basically simple collections of individual pictures or “frames”. The process of turning a model into a fully-lit 2D or 3D picture is called rendering.

The process of rendering is a highly compute-intensive task. Most workstations (indeed most computers) can effectively render only a single frame at a time. Since a video is composed of individual frames, if you rendered ½ of your frames on one computer and ½ your frames on a second computer, you could render all your frames in ½ the time. The more computers, or “nodes”, you have to render frames, the faster you can get your work done. There are now technologies that will allow you to render only a portion of a frame per node.

The processing power required to render even a few shots is significant, says George Johnsen, chief animation and technical officer at Threshold Digital Research Lab in Santa Monica, Calif. "In the visual effects business, there's no end to how many computers you can use," he says. A single shot can range from a few seconds to several minutes. Each second of film includes 24 frames, each containing up to 4,996-by-3,112 pixels in 32- or 64-bit color. Separate passes must be made for each object that requires rendering in the frame and for attributes such as texture, lighting and reflections.

"In the [upcoming] movie Foodfight!, there is a scene with 13,000 extras, and they all have animation cycles," says Johnsen. And artists often repeat the rendering process to improve quality. As many as 150 passes may be required - a frame can be processed on only one CPU at a time and takes 48 to 72 hours to complete, he says. ⁶

What exactly is a render farm? The term render farm in general means a collection of networked computers, called nodes, all rendering some part or all of a frame of animation. This is what computer scientists might call a loosely-coupled supercomputer made up of SP (Single Processor) or SMP (Symmetric Multi Processor) nodes. A render farm is simply a collection of computers connected via a LAN running special software.

What makes a render farm special are the three software components that interact with the hardware. Those software components are:

- The animators' applications (discussed above). Ideally these are integrated with a render manager directly in the UI of the content creation application so the artist can submit and see results of a network render job without switching contexts.

⁶ <http://www.computerworld.com/hardwaretopics/hardware/server/story/0,10801,96284p2,00.html>

- The rendering engine. The rendering engine takes information created by the animator and then "renders" it into cinematic, High Def, or TV quality frames. There is an instance of this software running on each render farm node, waiting for jobs to be sent to it by the Render Manager.
- The render manager (sometimes called a render queue manager). The render manager is a software system that schedules and distributes render jobs to render nodes. This is a fairly common task for all cluster based applications.

Render Engines

The render engine is a software package that is closely coupled to the content creation application. A render engine's job is to produce high quality, fully lit and shaded visuals. At a high level all a render engine does is take a bunch of files as input, then generates a JPEG (or some other format) as output. It may seem like a simple data transformation, but the transform contains some of the most complex mathematic algorithms known to computer science.

A render engine can operate individually, or in parallel with other instances of itself. Typically in a cluster one instance of a render engine will operate on each node in the render farm. It is not the render engine's job to manage the workload, or perform content management functions (those tasks are left to other applications).

The render engine runs both on the content creation workstation to render local jobs, and it runs on each node in the render farm. The render engine has to be tightly coupled to the content creation application running on the animator's desktop. Not only does the render engine have to understand how the content creation application visualizes data, the render engines are often directly integrated into the content creation application's user interface, so that an animator can have a render done without a UI context switch. Switching from a content creation app to some other app in order to do a render is very disruptive to the creative process. So in order to be as effective and efficient as possible most render engines (and render managers... see below) have some form of UI level integration with the market leading content creation applications. For example, Mental Ray has a plugin for Maya, allowing the two to be conveniently used together.

Flash Blue wants to use two render engines: Mental Ray, and Renderman, in addition to local workstation rendering on Maya. It is common in the industry to use multiple render engines, often choosing for each scene or task which tool is the best for this purpose.

Some render engines (e.g., LightWave), are fairly straightforward with one binary, and a simple installation and execution mode. However most other render engines have a core set of functionality and allow for extended functionality through the use of plugins. The plugins often perform specific functions such as rendering trees, hair, water effects, special lighting effects and even allow for custom shaders (plug-ins that help mathematically determine the correct shade or color for the various objects) code. These plugins have a render engine component, and a UI component, and need to be installed everywhere the render engine is installed. Unfortunately this means that we have to not only license and install the render engine on every node in our render farm, we also have to license and install the plugins on every node in our render farm.

Typically rendering plugins will increase the memory requirements of a render node. The industry has found that many render engines will fail on complex frames if there is not enough memory to process the frame. Also, these complex frames are more CPU intensive than your average frame. This combination has led to the concept of "fat" nodes: a few nodes in the render farm will be "fatter" than the rest. (This is sometimes called a "non-symmetrical" render farm.) In this case, the plugins will only be installed on a subset of

nodes. This also might be a good alternative if the plugins in use alter the hardware requirements of the render node.

Using a small number of “fat” nodes reduces the cost of the overall render farm by reducing average hardware costs and software license costs for plugins, but it will make the render process more complicated because frames that require the plugins or additional memory will have to be rendered only on the “fat” nodes. This requires manual intervention or sophisticated render management software that can understand the capabilities of individual render nodes but also determine what render plugins are required to render a frame and automatically manage render jobs appropriately.

Which of our applications will live on the render farm? From the list we have so far, Renderman, Mental Ray, Adobe After Effects, and Apple Shake are the candidates. However, Flash Blue is not planning to run After Effects on the render farm, so we can run only Linux on our render farm.

Luckily for the simplicity of our situation, Flash Blue has not specified any plugins. It’s possible that they really won’t need any – or, that they are merely too early in their development cycle to have discovered that they will. Either way, we asked the question, none were specified, so we’ll go ahead on the assumption that none will be required.

If we design our solution properly, though, we can always add “fat” render nodes later, to give us a non-symmetric render farm. It’s wise to design in flexibility anyway, since render requirements always seem to grow over time, technology changes over time, and we want our solution to be viable for the entire life of the project and beyond.

Render Management

Render management applications are tool sets for managing the process of distributing a “render” (a frame to be rendered) out to a farm of servers (the “render farm”). It is the primary job of a render manager to:

- Maintain a list of computers on a network (or networks) that can be used as compute resources for rendering requests
- Maintain state information about the render resources (online, offline, in-use, available)
- Collect all the data files in use by an artist, and move them or otherwise make them available to each node in the render farm
- Launch the render application(s) on render nodes
- Collect the results from each node and move or process them
- Maintain detailed logging and reporting of the render process
- Report when render has completed (through email, or other notification)
- Report when errors occur

Each render manager will provide these functions differently. Some render managers are integrated right into the content creation application, allowing animators to submit jobs and view the results without leaving the context of their work environment. This is the ideal situation. Almost all render managers provide some client UI that allows for the submission of jobs from the animation workstation.

Once a render job has been requested the render client typically bundles up all the files needed to render each frame in a scene into a single package, and the package is copied to some location from which each render node can read it. After processing the input, the render node then writes the output file into the same location, with a different name, and informs the render manager that it has completed the task.

Render management tools often come from the content management vendor. However, many of the Tier 1 houses have written their own render management applications because of the extreme complexity and the highly unique mix of process and technology found in each studio: for example, Lucas Film/ILM, Pixar, PDI have all written their own render manager. Pixar's render manager, Alfred, can in fact be purchased from them, and is particularly well-integrated with their render engine, Renderman. Some companies use standard cluster management applications like Platform Computing LSF. These tools should not be confused with cluster management packages like IBM's own CSM and xCAT. Some render management applications are also coming onto the market, like Pipeline FX.

Flash Blue has not specified a render manager in their list of applications. This is a gap!

CPU Architecture: Should we use Intel, Opteron, or Power for our render nodes? The current norm is Intel. For some rendering applications and in some customer benchmarks, Opteron has been shown to reduce rendering times by 60% or more compared to 32-bit Intel. It is believed that the bandwidth of the internal data path provides the benefit. There are few applications that run on Power architecture (including the new cell processor) today, but this powerful platform is well-suited to rendering – and now with Linux running on Power-based blades, it should be only a matter of timing before we get to see this become a very viable option.

The typical world-class animation studio has a 1,000 node or so render farm. This is a lot of servers, a lot of cables, switches, racks, power, and air conditioning. Even studios that are just building their first phase and buying far less for now, often have one eye out for their future expansion needs.

Blades are a very good fit for render farms because they combine the most computers per square foot, with aggregation of common functionality (power, networking, cabling) that can reduce the capital and operational costs of a loosely-coupled supercomputer. Have no doubt, a render farm is a super computer. The system used for visual effects on "Lord of the Rings" could easily have been used to do real-time simulation of weather and current patterns in the Pacific Ocean, calculate quantum behaviors of exotic materials fempto seconds after the big bang, or discover the process by which a particular protein is formed from a strand of mitochondrial DNA.

Generally content creation companies have very limited technological resources, including operational staff and physical machine room assets. The blade form factor scales very easily, and is much easier to manage than traditional desk-side or rack-mount systems. In addition, with IBM's BladeCenter technology, cost savings in the network, KVM (Keyboard Video Mouse), power and floor space increase with the size of the render farm. At a certain, currently unknown, scale the cost savings of blades over rack mount systems allow them to pay for themselves.

There are blades available today in many different CPU architectures, including AMD-based, Intel-based and IBM Power-based. They run Windows and Linux. No matter what combination of operating system and CPU architecture we want, we can do it within the blade form-factor.

Because of the price-performance of the current Opteron based systems, we will choose AMD-based blades for our render farm. This also keeps our render farm CPU architecture consistent with our workstations.

Operating System: As before, there are two primary choices, Windows and Linux, for the render nodes' operating system. Luckily for us, all of the applications that we want to place in the render farm run Linux, so we can create a Linux-only render farm.

Let's re-draw our picture, with the information we've gathered and the decisions we've made.

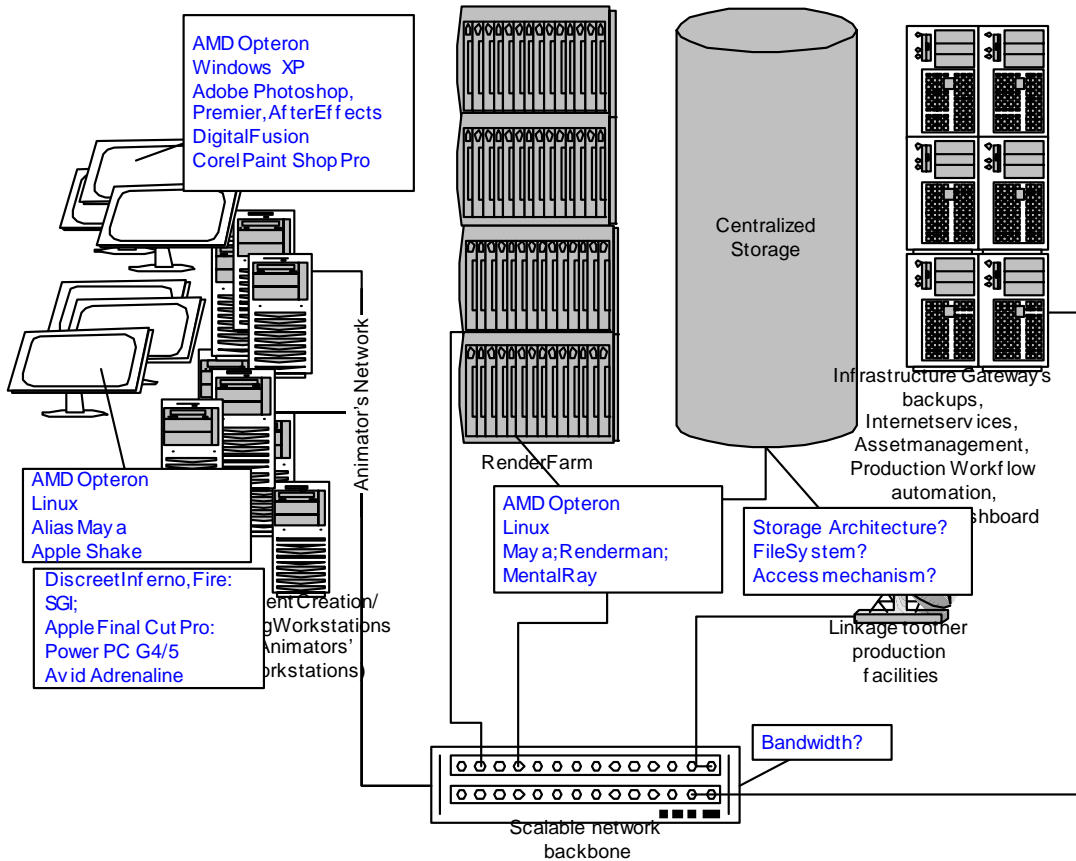


Figure 5: The Decisions So Far...

With our workstations and server decisions beginning to take shape, let's look next at the centralized storage components. Before we configure all our servers we must decide how we are going to connect all those render nodes to storage.

Centralized Storage

Rich content consumes massive amounts of storage. Also, as mentioned above, video editing requires high throughput disk I/O. For example, creating Treebeard, the talking tree in "The Two Towers", required roughly 80 GB of *textures* alone.⁷

In the past, this data was kept local to the animation workstations and the render farm nodes. This was mostly due to cost and complexity of SAN storage, and poor performance of NAS storage. Both those barriers are eroding. At the same time, the data requirements have been increasing, which makes local storage less practical.

“A single shot can require a terabyte of storage ... up from a few hundred megabytes a few years ago, while the work on a single film may generate in excess of a petabyte of data.”⁸

Clearly, having a petabyte of storage local to workstations is not a good idea. And with the recent move to even higher frame resolutions and higher frame rates, these requirements continue to escalate.

Historically the industry has shied away from traditional centralized (SAN-based) storage due to high capital and operational costs, with low perceived value over local direct-attached storage. Many customers in this industry view storage as a low-value commodity, which is interesting for an industry that uses the term “asset” to refer to digital content. Fortunately newer SAN storage offerings, and the introduction of low cost technologies such as high performance NAS and iSCSI, are changing the buying behavior of the industry.

Centralizing storage in a company creating rich visualizations has some non-obvious operational benefits. When assets are stored centrally they can easily be tracked, monitored, and reused. This leads to improved production efficiencies, which lead to shorter production times, and projects can be delivered in a more predictable time frame.

Centralized storage of the animation data (assets) is the natural and desired state of the industry. Centralization requires management, which in this industry is seen as a good thing. As of now, asset management has a long way to go to catch up with the tooling and management environments found in the software development world. Animation, pre-production and post production companies face the same set of problems, with regards to teams independently working on digital components which are, after several revisions, assembled into a final product.

At the moment all vendors have to offer are fairly crude asset management systems. Ideally artists and contributors would use a configuration management system, asset management system, or other such centralized repository. Unfortunately the current state of tools are not adequate to the task of managing the complexity of assets, volume of assets, and size of teams (can you say “business opportunity”?).

In many cases, studios use a file system instead of an asset management system, implement some naming convention and use a directory structure to organize and manage all the data and files that go into a production. Needless to say, this is inefficient, labor-intensive and error-prone. This is Flash Blue’s current approach; and, since we don’t currently have a compelling (in our opinion) alternative to offer them, we’ll accept this preference for now; however, we’ll

⁷ Ref: Intel Business Center Case Study: “Towering” Achievement: WETA Digital and Intel Architecture Create Effects for The Lord of the Rings Trilogy

⁸ “Blades, Camera, Action!” Robert L. Mitchell, ComputerWorld, October 4, 2004
<http://www.computerworld.com/hardwaretopics/hardware/server/story/0,10801,96284,00.html>

recommend that they use a tool to enforce file naming conventions, such as Data Frameworks.⁹ This can help reduce the difficulties of managing their file systems.

We include a file systems discussion whenever discussing storage systems, since the file system intermediates between the view that the user has of storage, and the real storage layout. The file system layer actually gives technologists a tremendous opportunity to do all sorts of wonderful things, such as put a network between the user and the disk drive, have a file span multiple disk drives and many, many other things.

We have several choices for our storage hardware architecture: we can choose Network Attach Storage, Direct Attach SAN, we can choose IP SCSI or we can choose a network file system or one of its successors, advanced file systems. We may even choose a combination of them.

Let's talk about each of these options, starting with storage options, then networked file systems and advanced file systems.

NAS

NAS has the right price point for this industry, but there are stability and performance problems inherent with inexpensive NAS. Traditional NAS vendors cannot provide the bandwidth required for production rendering.

SAN

If a studio has a rich budget SAN is the only way to go. Fibre attach SAN connectivity is relatively expensive compared to all other storage attachment methods. Currently every aspect of fibre-attached storage is expensive: the cables are expensive, the adapter cards are expensive, the SAN switch equipment is very expensive, the fibre SAN storage controllers are expensive. We also often need GBIC connectors for each link in the system, and these add considerable cost. At the moment, 2 Gb fiber is common, with 4 Gb bandwidth just becoming available.

iSCSI

SCSI over IP (iSCSI) has recently become an extremely viable alternative. The technology is maturing, it is cost-effective, and it uses networking skills that are more likely to exist in-house. However, since this is a fairly new technology in this market, there is little field experience and "rules of thumb" for how to configure these solutions in these situations.

iSCSI is a block based protocol on top of TCP/IP. A remote storage system presents itself to the host as if it were a local attached SCSI device. iSCSI is a compromise between the high performance of SAN and the ubiquitous connectivity of Ethernet and TCP/IP. Almost every modern operating system supports iSCSI natively, allowing virtually any node on a network to become either an iSCSI server or an iSCSI client.

Performance of iSCSI can be made to approach that of SAN if we dedicate multiple Ethernet interfaces to iSCSI in parallel. This is known as "channel bonding". Consider the number of Gigabit Ethernet interfaces a server can effectively handle. If we have a system with an I/O bus that can support two gigabit Ethernet adapters, we can theoretically approach the performance of a 2 gigabit fibre SAN link. If we extend this notion, with modern commodity PCI-X systems that can support multiple gigabit Ethernet cards, there is the potential to exceed the performance capacity of fibre SAN. In order for this to perform effectively another technology called Link Aggregation should be included. This allows us to combine multiple IP links, aggregate their bandwidth and present only one IP address for the combined

⁹ <http://www.dataframeworks.com>

set of links. Again, as with other environments, isolate SAN traffic to its own private LAN (or at the very least its own private VLAN).

Mixed SAN/NAS

Another way to provide SAN performance without the high cost is to create a mixed SAN/NAS architecture. The best way to describe this technique is to imagine 100 nodes needing access to SAN disk. Instead of attaching all of them via fibre, only attach 10 of them via fibre. Then, turn each of these 10 nodes into network file servers for 9 other nodes. This may sound strange, but it can be quite effective in an IBM BladeCenter environment where we have potentially 2 gigabit links. With 2 x 2gigabit links between the nodes inside the chassis this makes a promising economic compromise.

In addition, combinations of online, near-line and offline (tape) infrastructures can be an attractive way to reduce the costs of these kinds of infrastructures. We often see a combination of different storage architectures within a single studio.

Just say “No!” to transaction oriented storage systems.

IBM Shark and similar storage systems have excellent transactional performance, but these workloads are not transactional in nature. In general, IBM’s FAStT and iSCSI storage systems are appropriate for this industry.

Network File Systems

Sun’s NFS was not the first network file system, but for over a decade it was the most popular. Today NFS and the Microsoft-based standard CIFS are both quite common. These file systems provide ubiquitous low cost access to storage but at the cost of performance. All major Linux distributions come with the ability to serve and access NFS file systems, and CIFS file systems through SAMBA. Unfortunately NFS access on Windows is not straightforward and often requires commercial software.

Advanced File System(s)

Advanced file systems are a logical extension of the network file systems market. Where the network file systems allow a single server to act as a file server to a network of clients, the advanced file systems take this concept to the next level. They allow multiple file servers, attached to multiple storage controllers, to act as file servers to a collection of clients. In doing so, they hide the complexity of the underlying storage architecture from the clients, thus providing a level of “storage virtualization”. This also allows them to provide other benefits, such as providing a single namespace across multiple storage subsystems, and providing the clients with the aggregate performance of all the underlying storage subsystems.

There are two general architectures for these file systems:

- In one design, the file systems metadata is stored in specialized metadata servers. When a workstation wishes to access a file, it makes a request to the metadata server for the file. The metadata server replies, giving the physical location of the file. The workstation (via the advanced file system driver) requests the file from its real, physical location. Examples of this style of design are IBM SAN File System, Open-Source Lustre, and SGI CFXS.

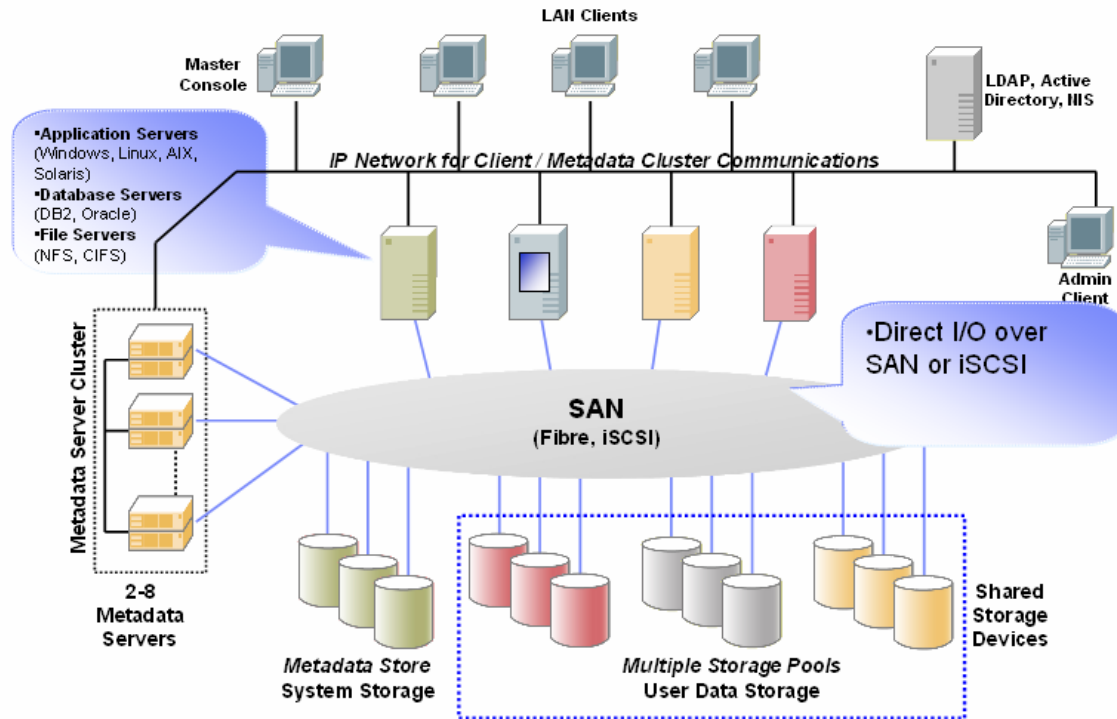


Figure 6: SAN/FS Architecture

- In the other major design, the storage subsystems are attached to a set of file servers, who mediate all requests for files. Metadata and data management are handled by the same servers. Here, when a workstation wishes to access a file, it makes a request to one of the file servers for the file, and the file server serves the file back to the workstation. Examples of this style are IBM Generalized Parallel File System (GPFS), and Polyserve Matrix Server.

Choosing a Storage and File Systems Design

What are the key factors that would cause us to use one of these over another?

An animation environment primarily deals with large numbers of files that must be moved between the render farm and storage, and to a lesser extent between the workstations and storage. Here, the metadata-server approach shines, since it allows each individual client to achieve roughly the performance of the underlying storage subsystems, but the metadata server is not involved in the actual data transfer.

A key in these environments is connectivity and support. Most of these file systems or storage solutions are more attuned to particular combinations of client and server operating systems. So, the first task is to collect the operating systems support requirements we have, and see if that eliminates any of our potential solutions. Alternatively, it may cause us to create a combination approach in our infrastructure.

Since our environment is animation-based, we'll lean towards a metadata-based solution, such as SAN/FS. Given that we also have heterogeneous attachment needs, we'll set up one of our file servers as an NFS server, in order to allow the SGI and Apple systems to attach.

For Non-Linear Editors, bandwidth requirements for each editor are high. Thus, the more NLEs that are expected in the environment, the more the storage environment will need to be

sized to take these into account. Especially if the NLEs are expecting to work on the same video stream at the same time (which can be supported using GPFS, for example), we will need the aggregate bandwidth of the underlying storage systems to fulfill requests from a single client. Here, we would tend to recommend a file system from the second approach.

“The actual requirement of a single video editor, production or broadcast station is quite simple, it must be continuously fed with one or more streams of video asset data to provide the input for the workflow. The challenge, specific to video data, is that the streams must retain a constant, deterministic flow rate and that the frames of the asset material must be delivered in order and in the correct time slots. This is referred to as isochronous transfer or delivery.

With a simple, stand alone system, this is relatively easy to deliver because there is only a single stream of material through the system at any time and the various factors combine to optimize the data flow in this environment, the situation becomes much more complex, however, when collaborative workflows are involved and when many systems, potentially performing different tasks, are connected to, and sharing asset material from, the same media asset pool.

There are two key aspects affecting the ability to retain an isochronous flow. Firstly there is the raw performance capability of the system, which is the area most often concentrated upon in sales and marketing literature. Raw performance has, and continues to increase massively and many systems, even quite basic ones, are now capable of delivering the baseline performance needed for media production suites, in a stand alone environment. The second factor, which comes more into play in complex environments, is determinism, affected significantly by latencies in the system. As the performance demands increase, and especially as the data flow becomes non-sequential, even though the raw performance may be adequate the confidence interval that the next frame will be delivered in time decreases due to the increase of instability and decrease of determinism, driven by the variety and variability of the factors influencing the data flow. This is crucial in a media environment, but of almost no significance in an IT world, where only the average flow rate of data is considered.”¹⁰

This is true both for storage bandwidth, and for networking.

On the other hand, not all NLEs support attachment to industry-standard file systems. If the studio is using, say, a proprietary NLE, it may be necessary to export the files from the file system to the local workstation, edit them there, and then return the edited files to the file system. This, naturally, makes the workflow more error-prone and time-consuming, but does reduce the performance requirements on the underlying storage subsystem – since the data must no longer be streamed to the NLE in a performance-sensitive fashion. However, editing, say, a 90 minute video this way is bordering on the impractical, due to the amount of storage that must be transferred back and forth. The industry is strongly moving away from this approach.

So, again, it’s likely that we’ll end up with a set of requirements with no single easy answer, and potentially with a hybrid design in order to satisfy them.

So What Will We Do Here?

Of the 5 NLE’s we’re expecting to support, all are capable of attaching to and reading files from SAN-based or NFS-attached storage, so we’re in fine shape there. We do have substantial bandwidth needs if our 5 NLE’s will edit streams at the same time. Therefore

¹⁰ “GPFS: Parallel access to data Advantages of parallel file systems in media environments”, Pete Hulme - Sagitta Performance Systems; Version 1.015 May 2003

we'll choose SAN as our storage architecture. We do not have a requirement from Flash Blue to have multiple systems editing the same video stream at the same time, so our metadata-based file systems approach is still appropriate.

However, animation and NLE's access storage in a dramatically different fashion. NLE's consume streams of high-bandwidth data, and experience jitter, hangs, and other effects if the next frame is not ready when needed. Rendering transfers smaller files in a batch fashion, and can tolerate delays – the results just take longer to create. Trying to deliver both kinds of data from the same storage infrastructure is fraught with performance problems. Even if solved once, these problems can easily recur when a small change occurs in the workload pattern of a single NLE.

Since we have requirements for both NLE's with their high-bandwidth, large file, performance-dependent requirements, and rendering, which moves many small files across the network in order to perform more "batch-oriented" work, we would be better off to separate the two needs into two separate pools of storage, each oriented towards one of these needs. This will simplify performance management significantly, and allow us to build a more cost-effective storage infrastructure with more reliable performance and easier problem resolution.

Given our choice of SAN/FS as our file system, we can manage this very simply, within our single namespace construct. We will simply define two filesets, each supported by a separate storage pool, with different storage policies associated with them. All individual frames and associated files will be stored in one fileset, in one storage pool. Once the individual frames are concatenated into sections of video, for example in MJPEG format, they will be stored into the other fileset, on a separate storage pool. This way, we can separate the I/O requests for video streaming from the small file transfers, optimize the network into separating the I/O, and still present a single filesystem to the artists.

All of our NLE's except for Avid can connect to and read from our centralized storage. However, the proprietary systems such as SGI do not have native file systems drivers available for our chosen file system, SAN/FS. They can, however, mount NFS file systems. So, we will have to provide NFS server access to the NLE storage. For Avid, we will have to virtualize our centralized storage, but we know how to do that too.

Networking

"The tie that binds, really chafes".

Binding all these various components together is the network. Video editing can easily consume all available network bandwidth, especially when you consider that to edit video one must stream video in both directions between centralized storage and editing workstations.

Network bandwidth is critical to the timely delivery of any production in this market. However, networking equipment is considered expensive compared to the perceived value it provides to the creative organization. These companies often find they grossly overestimate the capabilities of cheap network equipment, and grossly underestimate the amount and type of network activity required for timely production. We have seen, time and time again, companies have to re-purchase network equipment in the middle of a production:

Producer (day 1): What kind of network do we need to produce this 240 min of HD?

Techie (day 1): Based on careful analyses of the script and special effects work, we are going to need two huge 10 Gig routers and several hundred 2 Gig Ethernet ports, plus over a thousand 1G Ethernet ports.

Producer (day 1): Sounds good! Price it out! (Thinking to himself: Whatever, that took way too long, now I'm late for my lunch arrangements).

Techie (day 2): Our network will cost 8 million dollars. Our network vendor says we can start out slow with only a 6 million dollar investment and then add capacity as we need.

Producer (Day 2): (Gagging and coughing) You're kidding me right? It's just network junk! That network vendor must have seen you coming a mile away, you're getting suckered. We'll show them. I was in Fry's yesterday and I saw a bunch of networking gear, it was all under 100 bucks. The boxes said it was all high performance top of the line stuff. You go down to Fry's and buy that stuff.

Director, production manager, assistant producer (Day 180): Our production schedules are falling behind, our animators can't get work done, everything is "slow", we're going to miss our deadline, and then the distributor will pull our funding, then sue us for breach of contract.

Techie(Day 180): That networking equipment can't handle our traffic. It will support X but we need 100X.

Producer (Day 180) How much is it going to cost to fix the problem?

Techie (Day 180) 8 million dollars, and we have to just throw away all the equipment I bought at Fry's.

Producer: It's that or a day in court. Do it...

Unfortunately adequate bandwidth has been out of the price reach for most studios. Fortunately, like storage, competition has been driving the cost of networking equipment down to where animation companies can begin to afford 10 gigabit backbone systems.

Putting it All Together

Now, we've read the introduction, skimmed the big picture, and slogged through the gory details. On the way through, we've gathered information from the studio, and made a number of decisions about what to use for each of the components.

We'd captured some of this information on our high-level architectural diagram; below is our updated solution for Flash Blue.

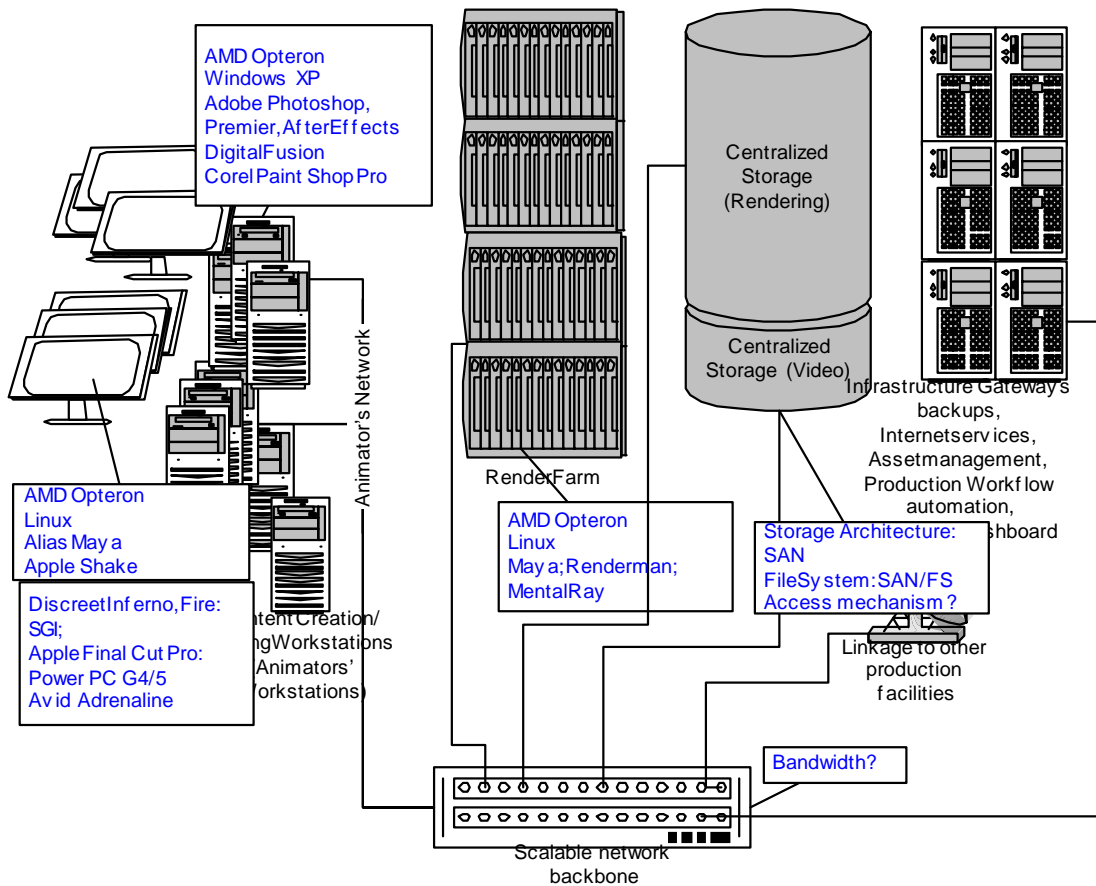


Figure 7: Proposed Architecture for Flash Blue

Now, we need to adjust this overall picture, based on how some of these components and decisions will interact with each other; then, we can move on to the next challenge, sizing.

Since we're using an advanced file system to access our storage, the easiest approach, from the perspective of the vendors of the file systems themselves, is to install the file system drivers on every potential "client" (whether animation workstation or render farm node – the file system regards anyone who wishes to read or write to them as a client). If licensing of the file system is by client, this naturally drives up the cost, potentially to the point of being unaffordable. In addition, the studio may not wish to install drivers on every client; or, it may not be technically feasible if the file system does not provide drivers for every O/S in the studio (and there are almost invariably a few oddball or proprietary systems for which it does not, or where installing the driver invalidates the vendor's warranty). This is our case, where our Avid and SGI Irix systems are not currently compatible with SAN/FS.

An alternate, common approach is to install the drivers on some subset of the nodes, and regard them as file servers to the remainder of the clients. Then, provided the oddball systems can use a standard protocol to access a remote file server (and almost all do), you've reduced your license costs and handled your oddball systems.

All major Linux distributions come with the ability to serve and access NFS file systems, and CIFS file systems through SAMBA. Unfortunately NFS access from Windows is not straightforward and often requires commercial software; and SAMBA is not yet as stable and scalable under Linux as it is in a Unix environment.

Our file system choice has led us to add a couple (or more) storage servers; in this case, we will need 2-3 metadata servers, and at least two NFS servers, to handle the file access and provide some level of redundancy. These servers are in addition to the number of servers we need inside our render farm. We cannot use these for rendering, as it will severely impact our storage performance and therefore our overall render farm performance, far in excess of the additional CPU you expect to exploit.

We decided on blades for our render servers and IBM BladeCenter for our render farm. We now have a choice on whether to connect the blades directly to storage, or use external 1U servers as storage servers. Current practice is to use 1U servers separate from the farm itself, and connect these storage servers to the render farm via networking. Currently, this ends up to be the most cost-effective and scalable fashion to provide the needed performance. While other solutions can work well with a small number of systems, they add complexity and quickly break down as the number of render farm nodes and needed file servers continues to increase.

How should we connect the BladeCenter to the outside world, out of the several options available? When we make this decision, we need to understand the type of traffic flowing between the external world and the blade. Best practice is not to mix “command and control” traffic, which has a large number of small packets, and storage traffic, which has a small number of large packets.

The most economically feasible option for a render farm that we expect to scale out significantly is the following:

- Connect one Gigabit NIC from each blade to one BladeCenter switch, for “command and control” and general traffic.
- Connect the other Gigabit NIC in each blade to a second, high-speed switch in the BladeCenter chassis, to be used for storage access. This provides the storage network isolation we’re looking for. This switch is then connected to 4 GB Ethernet connections, channel bonded together, shared for each 14 blades. For this, you need the high-performance Nortel or Cisco switch options in the BladeCenter. The Nortel or Cisco switch is then connected to the storage network backbone. Then, use a virtual LAN to isolate the traffic between the two – preferably with IPV6 with jumbo packets enabled.

With the new HS20 blades, we can have 4 x 1 GB Ethernet NIC’s for each blade, doubling the potential I/O performance into and out of each blade – but we will need additional network switches to use them.

What about back up for all this storage? Some studios choose not to back up, since they have multiple copies of the content on video tape or in other places. Others use a variety of cheap back up mechanisms; it is rare, in our experience, that a studio is willing to spring the cost of the kind of back up that a transaction-oriented business tends to spend. In reality, they see back up as less important, from a budget perspective, than networking ... and we’ve already discussed the challenges there!

Now, finally, we’re ready to move onto our last key question: how many of each of these components will Flash Blue need?

Sizing

Now, we've drawn our architectural diagram, and annotated it with relevant information: hardware architectures where known, operating systems preferred or in use, and applications. Now we're getting down to brass tacks.

Now our key remaining question is, how big is it?

Here, we address sizing each of the components:

- How many workstations will this animation studio need?
- How big should my render farm be?
- How do I size the individual render nodes?
- How much storage will this animation studio need?
- What storage bandwidth will this animation studio need?
- How do I size the centralized storage system?
- How do I size the network switches?

In general each of the major components can be sized separately, but many elements are interrelated – so answers to each question may need to be adjusted based on decisions made for neighboring components.

How many workstations will we need?

In our case, this is easy to calculate. Flash Blue is planning a total of 100 artists at peak. They tell us they only plan to have 5 NLE's. Each artist needs a workstation or an NLE; we'll assume that at peak, one artist will be working on each NLE, and so we'll need 95 other workstations. For at least some of these workstations, we'll be adding dual graphics screens. But, because we have used a single workstation base, we do not have to be specific about which workstations will be used for which applications, so this makes it easy; we can decide which software to install on which workstations very late in our project cycle.

How big should our render farm be?

First, recall the reason for a render farm in the first place. The point of a render farm is to *speed the process of rendering a production quality scene*. Note that the render farm does not necessarily speed the process of rendering a *frame*.

The general goal of a render farm is to render an entire scene in the length of time it takes to render a single frame. "HUH?" you say, this sounds so arbitrary. Think of it this way: the most granular thing we can do in this workload is render a single frame. That is, one computer can only render one frame at a time. (We can do sub-frame rendering, but it is expensive, complex, and not all rendering software supports it, and it complicates the process to no end. And, we can still use the following sizing method, and then multiply our results by the number of subdivisions per frame). This follows the general supercomputing goal of processing an entire work set in the time it takes to process a single symmetrical element of that work set. Once we've convinced ourselves that this is a worthy goal, sizing a render farm is simple math.

For CINEMATIC render farms: First, we must know the average number of frames per scene. In order to know this, we must often average the number of cuts per scene (because you may only know the average number of frames per cut, and the length of each cut is more regular than the average length of a scene.) In today's world a shot vary rarely lasts more than 3 seconds. A typical scene is between 3 and 9 shots. Emotional scenes have fewer, longer shots, action scenes have lots more, and the scenes at the beginning and end of a film are typically longer than the scenes in the middle and climax of the feature.

So, let's say a studio is doing feature animation and their average is 6 shots per scene, with an average shot length 1.5 seconds. It is an action movie using 28FPS (frames per second). The optimal starting size of the render farm, for this feature is $28 * (1.5 * 6)$ or 252 nodes.

This sounds like a large number. Is it reasonable? Let's look at an example:

For "The Two Towers", WETA Digital used 200 workstations and 450 dual-processor, Intel Xeon servers for rendering. Towards the end of production, they averaged 1,400 processors rendering at any given moment, by using artists' workstations as additional rendering stations. In total, the movie took over four million hours to render. [REF]

For our 90 minute long movie, we get the following calculations:

- Total Frames : 162,000 (= 90min x 60sec/min x 30frames/sec)
- Scenes : 90 (given; this is equivalent to an average of 1 scene per 1 minute)
- 1 scene = 60 sec x 30 frames/sec = 1,800 frames
- Therefore, to render all frames in one scene at the same time, it would take 1,800 processors.

We also know that each frame may take 3 hours (e.g., Toy Story¹¹), or 48-72 hours (e.g., Foodfight¹²) to render.

- Therefore, to render all frames in one scene at the same time, it would take 1,800 processors, for somewhere from 3-72 hours.

Now, frame rendering time is a very variable thing, depending on the complexity of the graphics, the number of things to be rendered, and many other factors.

Flash Blue tells us that they are expecting their graphics to be similar to that in Toy Story, so they expect to be closer to the 3 hour range.

If our budget allowed it, we could further reduce the total elapsed render time by doubling or quadrupling the number of processors, and using each node to render a partial frame. From here on, our processor calculations must be balanced against our budget and productivity needs. If we're trying to release our feature in an earlier season, it may be worth it to add more systems to ensure maximum productivity of our artists.

Flash Blue, on the other hand, wants to balance the size of their render farm with their budget. So, after some discussion, they conclude it's more practical for them if they expect to render one scene in the time it would take 4 frames to render; this has the added benefit of only needing one quarter as many render nodes.

So, based on these estimates and our discussion, we'll need a render farm in the order of 450 processors.

Since we're looking at using 2-way blades, that gets us down to 225 blades; or, just over 32 fully-loaded BladeCenters (at 14 blades per BladeCenter).

¹¹ Ref: <http://www.computer.org/computer/homepage/0202/ec/>

¹² Ref: <http://www.computerworld.com/printthis/2004/0,4814,96284,00.html>

How do we size the individual render nodes?

In general, we begin with a 2-way Intel server class CPU, 1 to 2 speed generations behind the cutting edge to keep costs down, and as much memory as the OS can address per process (on a 32 bit CPU that's 4 GB). "Fat" nodes are typically 4- or 8-way systems with lots of memory (16 GB).

Rendering is a highly integer math intensive process. Therefore memory bus speed, memory speed, and CPU integer performance are critical. NIC performance and OS performance for compute and network intensive operations secondarily impact performance.

This brings us to the goal of sizing an individual render node. The goal is to get everything we need to render a frame, AND THE FRAME, into memory. Some frames take up much more memory because there is lots more going on than simple lighting and physics. The goal is to prevent a render node from ever having to swap. Or even more abstractly, anything that takes the CPU away from calculating the color of a pixel is bad. If the render engine can run on a 64-bit platform we are in luck!

The memory will also be affected by the specific rendering software that the company wishes to use.

SCSI, ATA, SATA: what type of drive should be in each render node? Experience after experience shows that IT DOES NOT MATTER. Here is why. Once the OS boots and the render application loads, the local disk is rarely accessed (and if it is something is wrong with the render setup). Remember the output frame will only be around 2 MB, so even with 4 GB of memory in the CPU, there is plenty of room on disk for the OS, huge render application, and a gigabyte of application memory.

Our frames are relatively low resolution – 1920 x 1080 pixels – so the memory needed for just the completed frame is relatively small: around 2 MB per frame. So our emphasis will be on accounting for the other software. We have a pretty average set of requirements here, so we'll start with a basic AMD-based blade, and the cheapest internal disk.

How much storage will our studio need?

To support their film Flash Blue needs a storage infrastructure that will support rendering 90 minutes of high definition video, at 2 MB per frame, at 30 frames a second – a pretty standard requirement.

In terms of raw disk space required: with their resolution of 1920 x 1080 pixels, 32 bit color, 1 version of each frame will take:

$$2,073,600 \text{ pixels} \times 162,000 \text{ frames} \times 32 \text{ bits} = 1.22 \text{ TB}$$

So, the rendered IMAGES will take up just over 1 terabyte, leaving out sound, script, image source, metadata, fragmentation, RAID storage waste, client backup, etc. etc. But, they will want multiple iterations, and multiple layers: so we'll estimate that the final is 10% of working storage ... so we'll estimate that they will need a total of 12 TB. For a SAN, that's not bad; we can support that with a single low-end controller and SATA disks.

Much of the work will not be sizing the CAPACITY of the storage system but trying to derive a cost/aggregate bandwidth balance. So...

What storage bandwidth will our studio need?

For our animation work, the I/O is primarily large, block-based access to a our centralized storage system. The I/O is also not sustained. It only occurs during two times:

1. When the render node gets a job submitted, and
2. When the render node is writing its data out.

For the render farm, getting the I/O requirement is easy using another rule of thumb:

Each render node will need 5 MegaBytes per second throughput. This is easy to do with a single 1 Gigabit (watch the Bits/Bytes in these calculations) Ethernet adapter on each node.

So, take the number of nodes we need, multiply that by 5 megabytes per second and Poof! we have our storage I/O bandwidth requirement. So, using the 5 MB/second/node rule of thumb, our 225 node render farm would need 1.125 terabytes/second storage bandwidth. Although this seems a large number, we can provide this with a set of switches and an intelligent design.

This calculation assumes all our nodes are reading/writing at the same time, which would be worst case so we can feel pretty comfortable with this estimate. Even if the actual I/O is higher, it is likely we will not get all nodes in the cluster at peak I/O at the same time.

Our experience is that the render process is fairly complex and "flows" rather than "batch run". In many environments a render job will get started, some nodes will start rendering, the results will come from those renders, and cause other jobs to start on other nodes, while the original nodes move off and do something else. There is a Linux journal article¹³ that talks about how Lucasfilm divides up their cluster. Even small clusters of 30 nodes or less will get partitioned and allocated to various artists and projects.

Now, we have to add our NLE bandwidth. Each of our NLE's will consume about 5 MB/sec, and we have 5 of them. So, we'll need another 25 MB/sec to handle this load. Given that this is an extremely performance- and latency-sensitive I/O requirement, we will handle this load through a separate set of switches, rather than intermixing the I/O with other storage requests.

How do I size the centralized storage system?

Modern storage systems are a complex mix of hierarchical interconnected systems. Most storage systems can be broken down into two basic subsystems:

1. The storage controller, and
2. The disks.

Initially, a small number of disks may be physically placed in the storage controller; after this, additional disks, up to the maximum supported by the storage controller, may be installed in expansion units. These are sometimes called JBODs, or "Just A Bunch Of Disks".

There two important rules of thumb when sizing current disk systems for the Media & Entertainment industry:

1. Each disk can transfer about 5 megabytes I/O per second.
2. Each Gigabit NIC can only provide 100 megabytes I/O per second.

What this means, in general, is that we'll need at least the same number of disks as we have servers in our render farm, since each server can consume about the same bandwidth as each

¹³ <http://www.linuxjournal.com/article/6011>: you may need to register to read the details.

disk can transfer. This is normally far higher than the number of disks calculated to support the actual disk space requirement. Given the workload characteristics, caching does not generally help much for animation; it is rare for the same file to be read enough times in a short enough time for caching to make a significant impact on performance.

In general the rule of thumb for maximum speed is... more drives at a smaller drive size give you better performance.

Remember that our design uses two storage pools, one for the NLE's, and the other for access by the render farm. We will size each of these separately.

For our NLE's...

“Let it be assumed that an editing workstation requires a single video stream at 40 Mb/s ~ 5 MB/s, an Ethernet network can provide 100 Mb/s ~ 10 MB/s and SCSI can provide 40 MB/s.

These are realistic figures from experience and take into account some of the framing overheads discussed in the referenced paper. Now examine the behavior of the system.

- With one client attached, 5 MB/s is drawn from the storage, through the server and on to the client – no problem
- With two clients 10 MB/s is drawn from the storage to the server and 5 MB/s is passed over the Ethernet to each client. The Ethernet network is now saturated (passing data at it's maximum possible speed) and so no more clients can be added. N.B. In reality no professional design would ever plan to fully utilize and aspect of the system because the determinism factors already discussed would make the solution unstable and unreliable, as already discussed, this model is used purely for illustration.
- Extra network cards can be added to the server, each allowing connection of a further two clients, though this depends upon the availability of slots in the server and its ability to manage the extra resource requirements. Also adding cards decreases the reliability of the system (MTBF decreases geometrically with the number of components.) An alternative is to move the server and clients to a faster networking technology, but the same limitations still apply, just at a higher number.
- Once 8 clients are operating, the SCSI bus from the storage to the server becomes saturated and so this becomes the system bottleneck.
- Extra (or faster) SCSI technology can be deployed, as for the networking, but again this just moves the problem.
- At some point the internal PCI bus speed, processor capability or resource management capacity of the server itself becomes exhausted and the actual server becomes the bottleneck (in some cases the capability of the OS can be the limiting factor).”¹⁴

¹⁴ Sagitta Performance Systems, Langstone Technology Park, Langstone Road, Havant, Hants, PO9 1SA; www.sagitta-ps.com, GPFS Paper 08-05-03 v2.doc Page 8 of 22, 15/05/2003

So, our 5 NLE's will use about 25 MB/sec. One SAN storage controller can handle this load; so can 1 decent-sized file server; but, we must make sure we have a 1 Gigabit network switch to connect these NLE's to the file servers.

Applying our two rules of thumb, we get the following rough configuration:

- For our NLE's, we will support this with 2 Gigabit NIC's, attached via a switch to one storage controller; and we will provide around 3 TB of space behind the controller.
- To support the peak I/O for the render farm, we'll need around 250 disks, and 12 Gigabit NICs. We can support this with around 6 small file servers, each providing file server access to a storage controller and around 42 disks.

Because of the I/O bandwidth requirements, we place these servers outside the render farm, on 1U servers. The switches integrated into the BladeCenter chassis then gives us sufficient bandwidth to connect between the file servers and the blades.

We now have our first draft storage subsystem configuration.

Sizing the File System Servers

Whether we choose a metadata-driven file system or a GPFS-style system, each advanced file system adds servers to our overall mix. Even if a file server approach is taken, these file servers must also be added. These additional file system servers should be regarded as dedicated servers. It is a mistake to try to use them for rendering – it always lead to bad results!

So, we need to size our file servers, and add them to the overall list of servers in our configuration. For this purpose, we'll just state that our sizing shows that we'll need 3 additional servers to support Flash Blue file system.

Note that the number of advanced file system licenses we will need depend on the access method we use, and where the file system's drivers are installed – since most file systems require a license for every system on which the drivers are installed. For example, if we run the file system on each node in our render farm and the client workstations, we will need licenses for each of these and for your storage servers. On the other hand, if we expose the file system via a network file system such as NFS or SAMBA, we will reduce our license costs, but will add other performance and stability complexities.

Where are we going to place the file system servers? As with our file servers, we don't want to use blades; we'll place them on 1U servers external to the renderfarm. This way we simplify the network topology.

Some file systems run better on servers that handle threading well, due to the file system's workload characteristics. Therefore, they may run well on Power-based servers, and we expect it would benefit greatly from Intel servers that support hyperthreading.

So, we have now added a total of 10 servers to support our file system needs: 6 for our file servers for the render farm, 1 for our NLE file server, and 3 for our file system (metadata) servers. We must remember to add these to our total server needs!

What networking bandwidth will our studio need?

Connectivity of both creation and editing workstations to the centralized storage system is critical for the smooth and cost effective production of creative content. In addition, the render farm's connection to the storage system must be taken into account.

One issue with the animation network is the traffic tends to be fairly "bursty". That is, we find load spikes of two or three magnitudes greater than the average. Fortunately when these bursts occur they tend to be unidirectional, either read or write, to or from the SAN or render farm. Also, these bursts do not tend to require "sub-second response times". That is, although we need to make sure that all the data gets to where it's going in a "reasonable" amount of time, we do not need to make sure that we will always be able to accommodate the absolute peak load.

This, of course, does not apply to the NLE traffic, which must get there when expected. This may cause the network expert to recommend a separate network for the NLE traffic from that provided for other storage traffic.

If we look at the network requirements of just a small number of animation stations and a dozen or so render nodes, it does not seem like we'd need that much bandwidth, but it all adds up quick. For example if we use our rule of thumb that each render node requires 5 Megabytes per second I/O, 20 nodes are going to push a 1Gb network to its limits. (Keeping in mind that real world utilization of Ethernet typically can not exceed 60 or 70% and many seasoned network engineers will tell you that 40 to 50% utilization is more realistic.)

If we consider our 225 node render farm, we calculated a need for 1.125 TB/sec, if all our nodes are trying to load data at the same time. The usual expectation for networks is 1 Gigabit. To create a 1.125 TB/sec throughput, i.e. 11.25 terabits/second throughput, will require a significant switching infrastructure – or, an intelligent partitioning of the nodes and the workload. Although this is a peak calculation, we certainly don't want to design too far below that need, or we'll find our entire render farm productivity gated by our inability to move the data as needed.

These kinds of bandwidths are significantly greater than the switch backplane of most cheap switches. Therefore, multiple switches would likely be needed, and the design can very easily work (or, conversely, not work) based on how the switches are cabled.

Also note that high-speed networking solutions often used in High Performance Computing, such as Myrinet, generally are not appropriate for these systems. Myrinet provides high speed networking between servers; for these workloads, there is little to no inter-server traffic; the traffic is almost exclusively from the server to the storage and back again. Solutions that improve TCP/IP performance for the file servers, such as TCP/IP Offload Engines (also known as TOE cards) may however turn out to be cost-effective. Again, these probably do not provide sufficient benefit for the individual render nodes to make them valuable when used there.

A significant number (more than any other single source) of the "poor performance" issues we have seen has been as a result of network switch infrastructures.

We're not network experts. Our advice is: involve someone who is. And, be very, very careful.

Isolation

Laboratory and real world tests have shown that non-storage traffic can truly interfere with the performance all forms of NAS (iSCSI, NFS, CIFS). We must make every attempt to isolate storage traffic to its own network: keep it separated from "control" or "general traffic" networks. At the very least if the networking equipment supports VLAN, we'll isolate the storage traffic to its own network.

Jumbo Packets

Real world experience has proven that using the IP V6 “Jumbo Packets” feature can dramatically improve NAS/iSCSI performance for streaming applications, and applications reading and writing large files. Unfortunately the jury is still out whether jumbo packets provide benefit to rendering performance.

The Final Picture

Now, we’ve calculated the sizing for each of our architectural components, so we can add this information to our architectural picture. Here is how our solution for Flash Blue looks now.

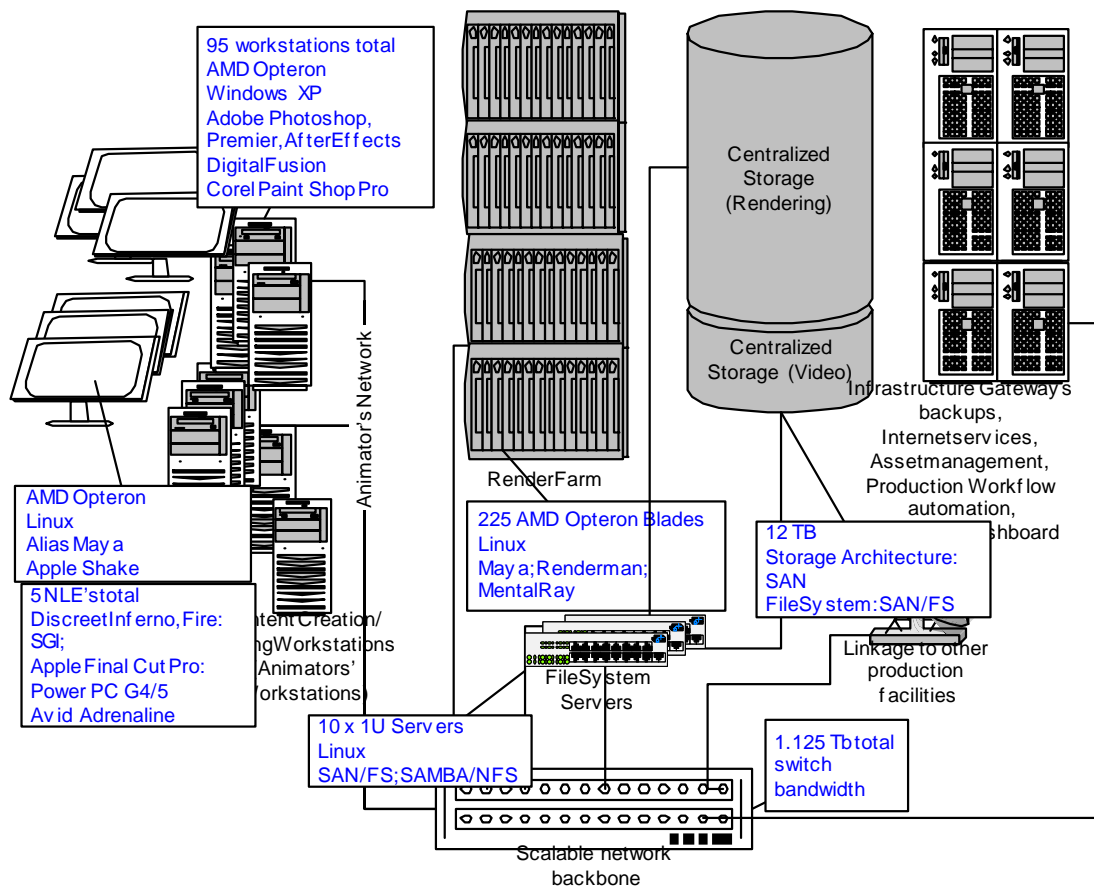


Figure 8: Final Flash Blue Architecture

Are we finished? Almost – but not quite. Now we need to help Flash Blue outline the rough schedule for their building out.

This is quite a large infrastructure to get to, from essentially nothing (they currently own a few file servers and some workstations, but no major IT). We suggest that they begin with a “Phase 1” – a smaller incarnation of this overall picture, sufficient to test the structure and the applications, and to support their first set of animators. Often, this can appropriately be performed during pre-production, since there will be a smaller team of artists working during that time. This can also, as they are building their first few fully animated frames, allow them to test their initial assumptions about the size of the infrastructure, and adjust or fine-tune their build-out. One of the nice things about this approach is that it can be added to in building blocks.

What are the key milestones? There are two sets we need to account for – the major milestones in creating the movie (which we already know, since the project has been greenlighted – and these dates were likely part of the negotiation with the funding studio); and the IT milestones, which are interdependent with these. We can work backwards from the studio project milestones to define the IT milestones.

At this point, we’ll just define the largest chunks of time we know we’ll need for the IT implementation:

1. System order, as the beginning milestone for system deliver time;
2. Installation: we’ll allow a substantial chunk of time here, in order to not just install the basic components (which we could achieve in a few days, if everything arrives and works well), but also to allow for bedding down of the systems, and for Flash Blue to experiment with the new setup.
3. Evaluation: monitor performance as we run test jobs and test renders, and begin to adjust our “build-out” accordingly. The changes may be the overall size of the render farm, but also our storage estimates and our networking setup must be monitored and adjusted.
4. Build-out, based on our adjusted estimates.
5. Full production.

We know that full production is when we’ll have all our artists working at full speed, and we certainly don’t want their productivity limited by lack of equipment. We also know that the evaluation period must start coincident with the beginning of what we’ll call “meaningful shot layout”; that is, once test shots have been laid out, and Flash Blue moves into cranking out shot layouts for the entire project. This is when real use of the render farm will begin to ramp up.

Here’s our summary of our high level project plan.

ID	Task Name	Start	Finish	Duration	Q205		Q305			Q405			Q106			Q206		
					Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May		
1	CompleteRequirements	5/16/2005	5/27/2005	10d														
2	OrderSystem	5/27/2005	5/27/2005	0d														
3	SystemDelivery	6/1/2005	6/30/2005	22d														
4	InstallRenderFarm	7/1/2005	8/11/2005	30d														
5	EvaluatePerformance, PlanExpansion	8/15/2005	9/9/2005	20d														
6	ExpandRenderFarm	9/12/2005	12/2/2005	60d														
7	Lay Out Shots	9/12/2005	10/7/2005	20d														
8	Hire Artists (20 - 30)	10/10/2005	11/4/2005	20d														
9	Begin Production	11/7/2005	10/20/2006	250d														

But, what size system do we want to begin with, for our evaluation period?

We want to build a sufficient infrastructure that Flash Blue can do an evaluation, begin productive work with the first group of artists hired, and confirm their configuration and sizing. In order to do that, they need at least a subset of each component in the infrastructure.

We’ll propose the following:

- One workstation for each artist. Quite a bit of work is likely to be done on the workstation before the render farm is needed anyway.

- A small render farm, of around 14 servers. This is sufficient that they can test the infrastructure, install the applications on a few servers each, and begin experimenting. They can even get some good productive work done, and validate their estimates for how long each render is likely to take.
- Two file servers, along with the attached storage and with 2 metadata servers. This will give them the “framework” of their storage infrastructure.
- They will need at least one NLE with local storage for a demo video. But since the majority of the NLEs work will not be needed until later in the production cycle, the second storage pool can be left till later.

Here is how our starting point for Flash Blue looks now.

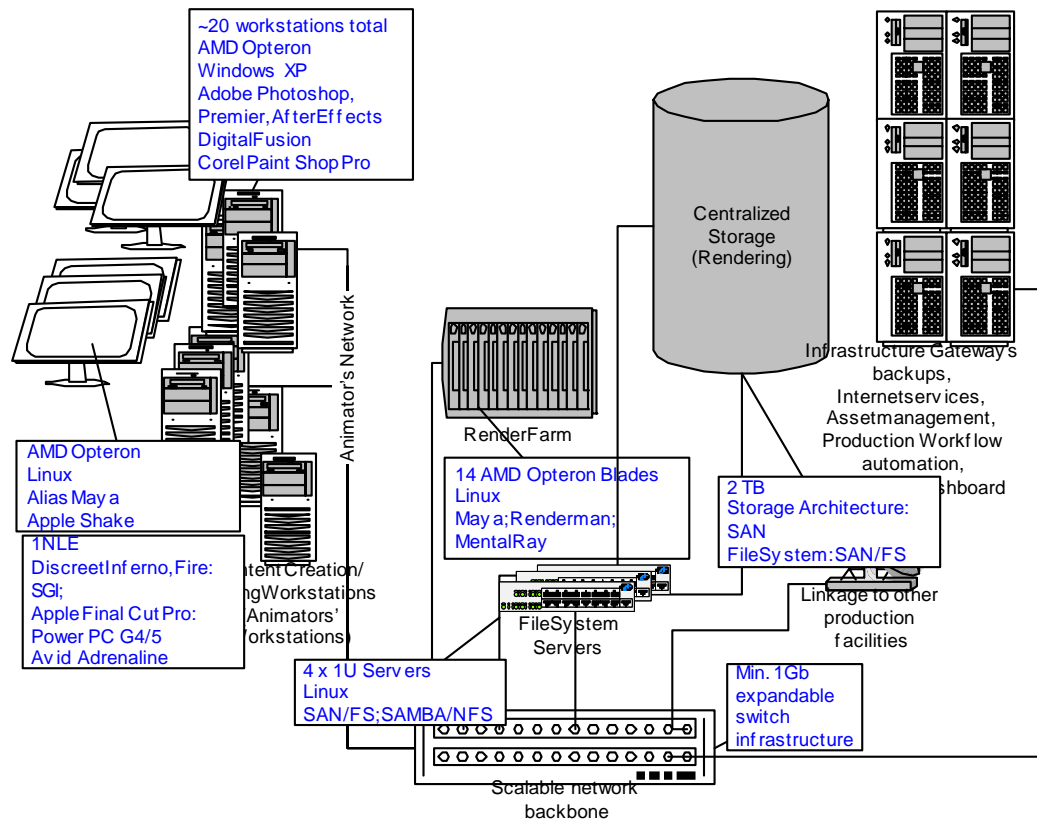


Figure 9: Initial Flash Blue Environment

And we can see how similar these two pictures are. By planning for our expected growth and designing a consistent starting point, we have covered both needs – for an entry point that allows us to validate our estimates, and a smooth growth path to our eventual full needs and beyond.

Now we really are done! We know what we’re planning to build, both in the short term, and the likely build-out (to be confirmed with real experiences); and we know when. We know that this solution meets all the needs Flash Blue has identified so far, and we know that the solution is flexible enough that it can handle adjustments, as they identify additional needs.

We’re greenlighted! Let’s build it!